Attorney Docket No.: 006570.P028 Utility Patent Application

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

FOR

UNIFIED LOGGING AND TRACING SERVICE EMPLOYED WITHIN APPLICATION SERVERS

INVENTORS:

GREGOR K. FREY
HEIKO D. KIESSLING
MIROSLAV R. PETROV
GEORGI G. MANEV
NIKOLA I. MARCHEV

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Boulevard, Seventh Floor
Los Angeles, California 90025-1026
(303) 740-1980

EXPRESS MAIL NO.

EV306655251US

UNIFIED LOGGING SERVICE EMPLOYED WITHIN APPLICATION SERVERS

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] Embodiments of this invention generally relate to a unified logging and tracing facility. More particularly, an embodiment of the invention relates to a unified logging and tracing service employed in an application server.

Description of Related Art

[0002] Java 2 Enterprise Edition (J2EE) is a tool commonly used in software development today. Generally, J2EE reduces the costs and complexity associated with developing multi-tier enterprise services. Another advantage of J2EE is that it can be relatively rapidly deployed and enhanced as the need arises. J2EE is currently used in many large-scale application development projects for these reasons.

[0003] As application development projects grow larger, logging becomes increasingly more important. Logging can be a very useful debugging tool to isolate problems, for example, by tracking each stage of the application being evaluated.

[0004] **Figure 1** illustrates a logging mechanism 100 in accordance with the prior art. The mechanism 100 utilizes a logger module 102 to receive logging messages. The logger module 102 provides the logging messages to a handler

104, which directs the logging messages to a formatter 106. The handler 104 determines where the logging messages should be sent. The formatter 106 may then provide human-readable data to a user based on the destination determined by the handler 104.

[0005] A major issue with current logging solutions is the determination of how and when messages are logged. The problem is exasperated when a development project involves many modules or developers. This problem is further complicated when different interfaces and/or services are used across the development project modules.

SUMMARY OF INVENTION

[0006] Disclosed are novel and efficient apparatus, methods, and systems for unified logging and tracing, for example, employed within application servers. In one embodiment of the invention, an application programming interface (API) provides configuration and/or viewing for both tracing and logging.

[0007] According to an embodiment of the invention, a unified logging and tracing system is disclosed. The system includes: one or more log controllers to communicate with an application or program module, represent source data, each of the log controllers receiving one or more messages; a log manager coupled to the log controllers to manage the log controllers; and one or more logs to represent one or more corresponding destinations for the received messages.

[0008] In one embodiment of the invention, the unified system includes a formatter coupled to each of the logs to determine a format of the received message prior to publication.

[0009] In another embodiment of the invention, the message includes severity information corresponding to that message.

[0010] In a further embodiment of the invention, the severity information is inherited.

[0011] In yet another embodiment of the invention, the inherited severity is restricted.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The appended claims set forth the features of the invention with particularity. The embodiments of the invention, together with its advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings in which:

[0013] **Figure 1** illustrates a logging mechanism 100 in accordance with the prior art;

[0014] **Figure 2** is a block diagram of a computer system;

[0015] **Figure 3** illustrates an exemplary logging tool 300 in accordance with an embodiment of the invention;

[0016] **Figure 4** illustrates an exemplary hierarchical configuration 400 for locations in accordance with an embodiment of the invention;

[0017] **Figure 5** illustrates an exemplary hierarchical configuration 500 for categories in accordance with an embodiment of the invention;

[0018] **Figure 6** illustrates an exemplary hierarchical configuration 600 with restricted inheritance in accordance with an embodiment of the invention;

[0019] **Figure 7** illustrates an exemplary class diagram of a logging API 700 for the logging tool 300 of **Figure 3** in accordance with an embodiment of the

invention; and

[0020] **Figure 8** illustrates an exemplary graphical user interface (GUI) 800 in accordance with an embodiment of the invention.

DETAILED DESCRIPTION

[0021] **Figure 3** illustrates an exemplary tool 300 in accordance with an embodiment of the invention. The tool 300 includes a log manager 302 coupled to one or more log controllers 304 (one shown in **Figure 3**, but the "N" markings in **Figure 3** indicate possible presence of N items). The log manager 302 manages the log controllers 304. In one embodiment of the invention, the log manager 302 is part of the kernel of the J2EE Engine and is started at system startup. The log manager configures the logging controllers for the kernel (e.g., by reading the configuration specified in the file *log-configuration.xml*). The log manager may also provide support for storing system critical logs in a database.

[0022] In an embodiment of the invention, a log controller 304 may receive messages from executing application. The log controller may be defined as a Java class having two subclasses or modules such as "category" to generate log messages and "location" to generate trace messages. Tracing involves the reconstruction of the control flow of an executing application, which may be used during program development and testing or problem detection, in productive systems (alternative to debugging). Tracing may be switched off during normal operation. Moreover, trace messages may be emitted to locations, which describe delimited code areas such as packages or classes.

[0023] Logging may be used to record normal or exceptional events that occur during program execution and this may be switched on during normal operation. The log messages may be emitted to categories, which describe

distinguished problem areas such as database problems or security auditing. In an embodiment of the invention, log messages are also visible in a trace.

[0024] The source, for trace messages, may correspond to a hierarchical Java naming package or class such as:

com.sapmarkets.usermanagement.util.xxx

com.sapmarkets.usermanagement.util.db.xxx

[0025] The source, for log messages, may correspond to a logical name defined by a developer such as:

/System/Infrastructure/Dev/UM/DB/xxx

/System/Infrastructure/Dev/UM/Network/xxx

[0026] In an embodiment of the invention, logs typically are addressed to an administrator of a running system while traces typically are addressed to a developer and/or support organization. Thus, the developer may decide whether a message is a log message or trace message.

[0027] The tool 300 further includes a log (or destination) 306 coupled to each of the log controllers 304. The log 306 may represent the destination to which the message received by the respective log controller should be sent (or otherwise published). The log 306 is coupled to a formatter 308, which determines the format of the message to be published.

[0028] In one embodiment of the invention, a message passed through the tool 300 may include a log record or potion thereof. The log record may be a

structure that holds at best a portion of a message and data corresponding thereto. The log record may include severity information regarding the message, for example, to indicate the importance (severity) of the message. A severity threshold may be set and only a message with an assigned severity that is more severe than this threshold may be recorded and/or published.

[0029] In one embodiment of the invention, there are a number of severities, for example, six, provided as constants: Severity.DEBUG (e.g., for debugging purposes), Severity.PATH (e.g., to indicate a path severity, for example to keep track of a path), Severity.INFO (e.g., to indicate an informational severity), Severity.WARNING (e.g., to indicate a warning severity), Severity.ERROR (e.g., to indicate an error), and Severity.FATAL (e.g., to indicate a fatal error). To disable writing or logging, a severity threshold of Severity.NONE may be utilized.

[0030] It is also envisioned that severities may be inherited, e.g., to reuse settings in hierarchies of code areas and message categories. For example, names of log controllers may be used to build implicit hierarchies. Locations may be named similar to qualified Java package and class names. For example with respect to the hierarchical configuration 400 of **Figure 4**, a parent 402 passes on its severities to children 404 and 406. Additionally, **Figure 5** illustrates an exemplary hierarchical configuration 500 for categories where categories are named in a manner similar to naming conventions used in hierarchical file names. In the configuration of **Figure 5**, the parent 502 passes on its severities to children 504 and 506.

[0031] **Figur 6** illustrates an exemplary hierarchical configuration 600 with restricted inheritance in accordance with an embodiment of the invention. The configuration 600 illustrates the restriction of the hierarchical configuration 400 of **Figure 4**. The child 404 has its own child 602.

[0032] As illustrated in **Figure 6**, configurations such as severity settings or logs may be passed on from parents to children unless overridden. In one embodiment of the present invention, the following logs permit restriction of inheritance:

Private logs, specified with *addPrivateLog*, may be written according to the severity setting of the parent instead of the child; and

Local logs, specified with *addLocalLog*, may be written when the parent itself is called.

[0033] Additionally, the inheritance of severity settings may be controllable via a range from a minimum to a maximum severity specified with:

<Controller>.setMinimumSeverity(<Value>); and

<Controller>.setMaximumSeverity(<Value>).

[0034] **Figure 7** illustrates an exemplary class diagram of a API 700 for the tool 300 of **Figure 3** in accordance with an embodiment of the invention. The API 700 includes a log manager 302, a log controller 304, a log 306, and a formatter 308. The log controller 304 has two subclasses or modules, category 702 and location 704 (such as discussed with reference to **Figure 3**). The log 306 has a subclass or module stream log 706 to enable writing to an arbitrary

output stream. The stream log 706 may further have two subclasses or modules: file log 708 to write to a file or rotating file set and console log to write to a console or system error log (such as *System.err*).

[0035] In one embodiment of the invention, the formatter 308 has three subclasses or modules: (1) a trace formatter 714 to provide a human-readable, configurable format specifically for traces that are not read with a viewer (such as the one discussed with respect to **Figure 8** below); (2) a list formatter 712 to provide a compact format of hash-separated message fields for processing in a log viewer (such as the one discussed with respect to **Figure 8** below); and (3) an extensible markup language (XML) formatter 716 to provide an XML-based format suitable for further processing in foreign applications.

[0036] **Figure 8** illustrates an exemplary graphical user interface (GUI) 800 in accordance with an embodiment of the invention. The GUI 800 includes a service selection area 802, which permits a user to select the provided services such as "log viewer" to view logged information in accordance with the techniques discussed herein (e.g., Figs. 3-7).

[0037] After selecting the log viewer option in 802, the user functions defined by separate may be provided with an interface 804, which permits the user to elect between different tabs. The tabs include a control descriptor tab 806 to manage the descriptors for the logging (such as the options discussed with reference to Figs. 3-7), a properties tab 808 to view and change properties associated with the log viewer, and a runtime tab 810 to illustrate logged data for

a given application (812), machine (814), and/or a task (816). The logging data regarding the selected runtime items may then be viewed in a portion (818) of the GUI 800. This data may include the identity of the thread, source, severity, message, message identity, date, time, location, and the like (such as illustrated in portion 818).

[0038] In one embodiment of the invention, an application programming interface (API) provides unified configuration and/or viewing for both tracing and logging. An embodiment of the invention takes advantage of similarities between tracing and logging, and of synergistic effects of handling both – while not neglecting differences such as: path messages for traces; cross-referencing between traces and logs; or language-independent messages for logs. It is envisioned that a unified approach to logging and tracing is configurable for different output formats and destinations, is highly controllable (e.g., uses controlled inheritance to reuse settings in hierarchies of code areas and message categories and/or allows relative control of message writing, for example enabling extended tracing for a classified subset of trace calls).

[0039] It should be appreciated that reference throughout this specification to "one embodiment" or "an embodiment" means that a particular feature, structure or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Therefore, it is emphasized and should be appreciated that two or more references to "an embodiment" or "one embodiment" or "an alternative embodiment" in various portions of this specification are not necessarily all referring to the same embodiment.

Furthermore, the particular features, structures or characteristics may be combined as suitable in one or more embodiments of the invention.

[0040] Similarly, it should be appreciated that in the foregoing description of exemplary embodiments of the invention, various features of the invention are sometimes grouped together in a single embodiment, figure, or description thereof for the purpose of streamlining the disclosure aiding in the understanding of one or more of the various inventive aspects. This method of disclosure, however, is not to be interpreted as reflecting an intention that the claimed invention requires more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed embodiment. Thus, the claims following the detailed description are hereby expressly incorporated into this detailed description, with each claim standing on its own as a separate embodiment of this invention.

[0041] While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive, and that the embodiments of the present invention are not to be limited to specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art upon studying this disclosure.

[0042] In the above description, numerous specific details may be set forth in

order to provide a more thorough understanding of various embodiments of the present invention. It will be appreciated, however, to one skilled in the art that the embodiments of the invention may be practiced without such specific details, based on the disclosure provided. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate functionality without undue experimentation.

[0043] Various embodiments of the invention have been described above. The various embodiments may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor or a machine or logic circuits programmed with the instructions to perform the various embodiments. Alternatively, the various embodiments may be performed by a combination of hardware and software.

[0044] Various embodiments of the invention may be provided as a computer program product, which may include a machine-readable medium having stored thereon instructions, which may be used to program a computer (or other electronic devices) to perform a process according to various embodiments of the present invention.

[0045] The machine-readable medium may include, but is not limited to, floppy diskette, optical disk, compact disk-read-only memory (CD-ROM), magneto-optical disk, read-only memory (ROM) random access memory (RAM), erasable programmable read-only memory (EPROM), electrically erasable

programmable read-only memory (EEPROM), magnetic or optical card, flash memory, or another type of media/machine-readable medium suitable for storing electronic instructions. Moreover, various embodiments of the present invention may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer to a requesting computer by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem, network connection, a wireless link, and the like).

[0046] **Figure 2** is a block diagram of a computer system. The computer system includes one or more processors 202-206. The processors 202-206 may include one or more multi-threaded processors having multiple threads or logical processors, and may be capable of processing multiple instruction sequences using their multiple threads, for example, concurrently. Processors 202-206 may include one more microprocessors, micro-controllers, field programmable gate arrays (FPGA), application specific integrated circuits (ASIC), central processing units (CPU), programmable logic devices (PLD), and similar devices that access instructions from system storage (e.g., main memory 216), decode, and execute those instructions by performing arithmetic and/or logical operations. Processors 202-206 may also include one or more internal caches (not shown) and a bus controller 222 to direct interaction with the processor bus 212. Processor bus 212, may be used to couple the processors 202-206 with the system interface 214. Processor bus 212 may include a control bus 232, an address bus 234, and a data bus 236.

[0047] System interface 214 (or chipset) may be connected to the processor bus 212 to interface other components of the system 200 with the processor bus 212. For example, system interface 214 may includes a memory controller 218 for interfacing a main memory 216 with the processor bus 212. The main memory 216 typically includes one or more memory cards and a control circuit (not shown). System interface 214 may also include an input/output (I/O) interface 220 to interface one or more I/O bridges or I/O devices with the processor bus 212. For example, as illustrated, the I/O interface 220 may interface an I/O bridge 224 with the processor bus 212. I/O bridge 224 may operate as a bus bridge to interface between the system interface 214 and an I/O bus 226. One or more I/O controllers and/or I/O devices may be connected with the I/O bus 226, such as I/O controller 228 and I/O device 230, as illustrated. I/O bus 226 may include a Peripheral Component Interconnect (PCI) bus or other type of I/O bus.

[0048] System 200 may include a dynamic storage device, referred to as main memory 216, or a random access memory (RAM) or other memory coupled to the processor bus 212 for storing information and instructions to be executed by the processors 202-206. Memory 216 also may be used for storing temporary variables or other intermediate information during execution of instructions by the processors 202-206. System 200 may include a read only memory (ROM) and/or other static storage device coupled to the processor bus 212 for storing static information and instructions for processors 202-206.

[0049] Memory 216 may include a wide variety of memory devices including

read-only memory (ROM), erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), random access memory (RAM), non-volatile random access memory (NVRAM), cache memory, flash memory, and other memory devices. Memory 216 may store program modules, objects, images, data structures, program data, and other modules that perform particular tasks or implement particular abstract data types that facilitate system use.

[0050] I/O device 230 may include a display device (not shown), such as a cathode ray tube (CRT) or liquid crystal display (LCD), for displaying information to an end user. For example, graphical and/or textual indications may be presented to a user on the display device. I/O device 230 may also include an input device (not shown), such as an alphanumeric input device, including alphanumeric and other keys for communicating information and/or command selections to a processor. Another type of user input device includes cursor control, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to the processors 202-206 and for controlling cursor movement on the display device.

[0051] System 200 may also include a communication device (not shown), such as a modem, a network interface card, or other well-known interface devices, such as those used for coupling to Ethernet, token ring, or other types of physical attachment for purposes of providing a communication link to support a local or wide area network, for example. Stated differently, the system 200 may be coupled with a number of clients and/or servers via a conventional

network infrastructure, such as a company's Intranet and/or the Internet, for example.

[0052] It is appreciated that a lesser or more equipped computer system than the example described above may be desirable for certain implementations. Therefore, the configuration of computer system 200 will vary from implementation to implementation depending upon numerous factors, such as price constraints, performance requirements, technological improvements, and/or other circumstances.

[0053] It should be noted that, while the embodiments described herein may be performed under the control of a programmed processor, such as processors 202-206, in alternative embodiments, the embodiments may be fully or partially implemented by any programmable or hard coded logic, such as field programmable gate arrays (FPGAs), transistor-transistor logic (TTL) logic, or application specific integrated circuits (ASICs). Additionally, the embodiments of the invention may be performed by any combination of programmed general-purpose computer components and/or custom hardware components. Therefore, nothing disclosed herein should be construed as limiting the various embodiments of the present invention to a particular embodiment wherein the recited embodiments may be performed by a specific combination of hardware components.